vCon
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2026

T. McCarthy-Howe
VCONIC

4 June 2026

vCon Generation Provenance
draft-howe-vcon-provenance-00

Abstract

This document defines a "provenance" extension for Virtualized Conversations (vCon) that records how a piece of generated content was produced by a generative model: the model and provider, the decoding parameters (for example temperature and top_p), the prompt or a hash of it, the vCon elements that were given to the model as input, and a hash of the output. The record is carried as a named provenance member on the analysis object it describes (and, for machine-generated dialog, on the dialog object), so that an analysis has a real, named home for "how this was generated" rather than overloading the analysis body or schema.

The extension is a Compatible vCon extension. It introduces no new top-level fields and does not alter the semantics of existing ones. Because the provenance record binds an output to its model, prompt, and inputs by hash, a signed vCon, or a SCITT transparency receipt over it, can attest to a verifiable derivation: not only that the analysis exists, but how it came to be.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-howe-vcon-provenance/.

Discussion of this document takes place on the vCon Working Group mailing list (mailto:vcon@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/vcon/. Subscribe at https://www.ietf.org/mailman/listinfo/vcon/.

Source for this draft and an issue tracker can be found at https://github.com/vcon-dev/draft-howe-vcon-provenance.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

A growing share of the content carried in a vCon [I-D.draft-ietf-vcon-vcon-core] is produced by generative models: summaries, classifications, sentiment scores, extracted entities, translated or redrafted text, and synthetic dialog turns. These land in the vCon analysis[] array (and occasionally in dialog[]), each with a vendor, a product, and a schema. What is missing is a place to record how the content was generated - the prompt, the sampling parameters, and the inputs the model actually saw.

Today that information either goes unrecorded, or is smuggled into the analysis body (mixing it with the analysis result), or is encoded into the schema token (overloading a field meant to identify a data format). Neither is interoperable, and neither lets a consumer answer the questions an auditor or a transparency log asks: which model produced this, under what parameters, from what prompt, over which conversation content?

Generation provenance is a small, bounded set of facts with a clear owner: the object whose content was generated. This document defines a Compatible vCon extension (Section 2.5 of [I-D.draft-ietf-vcon-vcon-core]) that gives that set of facts a named home - a provenance parameter on the analysis object, and on the dialog object for machine-generated dialog - and registers it through the object parameter registries that the core specification already provides.

The motivation is the same as for content provenance in media [C2PA]: as automated decision-making is increasingly subject to regulation [EU-AI-ACT] and risk frameworks [NIST-AI-RMF], the derivation of a generated artifact becomes part of the record that must be retained and, where required, attested.

1.1. Relationship to the Agent Session Extension

This extension and the Agent Session extension [I-D.draft-howe-vcon-agent-session] address two different scopes and are complementary:

* A provenance record describes a single act of generation - one model call that produced one analysis or one dialog turn. It is lightweight and attaches directly to the object it explains.

* An agent_session record describes a whole autonomous run - a multi-step trace of tool calls, results, and reasoning - carried as a structured analysis body conforming to the Verifiable Agent Conversations schema.

Most generated content in a vCon is the result of a single model call, not an agent run; for that common case, this extension is sufficient on its own. Where both apply, they layer: an agent_session analysis entry MAY itself carry a provenance member describing the model and parameters of the run, and individual generated analyses within or alongside a session MAY each carry their own provenance. The two extensions share the same model-identity vocabulary (vendor/provider, product/model name) so that the records agree.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

All timestamps in vCon documents conforming to this extension are formatted as Internet date and time strings per [RFC3339], matching the requirement in [I-D.draft-ietf-vcon-vcon-core].

2.1. Core Terms

Generative Model: A software system, typically a large language model, that produces content in response to a prompt and a set of decoding parameters.

Generation: A single invocation of a generative model that produces one unit of content placed in a vCon (one analysis entry, or one dialog turn).

Provenance Record: The structured object, defined in this document, that records how a generation was produced. Carried as the provenance parameter of the object whose content it describes.

Prompt: The full input presented to the model for a generation, including any system instructions, templates, retrieved context, and the rendered user request.

Input Reference: A pointer from a provenance record to a vCon element (a dialog, analysis, or attachment entry) that was supplied to the model as part of the prompt, optionally bound by a content hash.

Content Hash: A hash token in the sha512- form defined for the content_hash parameter of [I-D.draft-ietf-vcon-vcon-core] - the algorithm name, a hyphen, and the base64url encoding of the digest.

Compatible Extension: A vCon extension that introduces additional data without altering the meaning or structure of existing elements, as defined in [I-D.draft-ietf-vcon-vcon-core].

3. Extension Classification and Registration

The provenance extension is a *Compatible Extension* as defined in Section 2.5 of [I-D.draft-ietf-vcon-vcon-core]. It:

* Introduces no new top-level fields.

* Defines a new provenance parameter on the analysis object and on the dialog object, registered through the corresponding object parameter registries of [I-D.draft-ietf-vcon-vcon-core].

* Can be safely ignored by implementations that do not support provenance processing; the analysis or dialog entry remains well-formed and its content is unchanged.

* Does not require listing in the critical parameter. The provenance parameter is descriptive metadata; an unaware consumer that ignores it still reads the generated content correctly.

This document defines the "provenance" extension token for registration in the vCon Extensions Names Registry:

* *Extension Name*: provenance

* *Extension Description*: Generation provenance for content produced by a generative model (model, decoding parameters, prompt, inputs, and output hash), carried on the analysis or dialog object it describes.

* *Change Controller*: IESG

* *Specification Document*: This document

vCon instances that include any provenance parameter SHOULD include "provenance" in the extensions array.

4. The Provenance Object

A provenance record is a JSON object placed as the value of a provenance parameter on the object whose content it describes. The same object structure is used wherever provenance appears.

4.1. Placement

The provenance parameter MAY appear on:

* An *analysis object* (Section 4.5 of [I-D.draft-ietf-vcon-vcon-core]). This is the primary placement. The record describes how that analysis entry’s content was generated.

* A *dialog object* (Section 4.3 of [I-D.draft-ietf-vcon-vcon-core]), when the dialog turn itself was produced by a generative model (for example a synthesized agent reply or a generated message). The record describes how that turn’s content was generated.

A provenance parameter describes exactly the object it is a member of. It MUST NOT be used to describe a different element; input relationships to other elements are expressed through the inputs array (see below), not by placement.

4.2. Structure

The provenance object contains the following members.

4.2.1. Required Members

* *model* (object, REQUIRED): Identifies the generative model.

  - *vendor* (string, REQUIRED): The organization providing the model (for example "anthropic", "openai", "google"). When the record is on an analysis object, this SHOULD equal the analysis vendor parameter.

  - *name* (string, REQUIRED): The model identifier (for example "claude-opus-4-8"). When the record is on an analysis object, this SHOULD equal the analysis product parameter.

  - *version* (string, OPTIONAL): A more specific model or weights version, when distinct from name.

* *generated_at* (string, REQUIRED): An [RFC3339] timestamp for when the generation was performed.

4.2.2. Optional Members

* *parameters* (object, OPTIONAL): The decoding and sampling parameters used for the generation. Member names are not constrained by this document; implementations record the parameters they actually set. Commonly used members include temperature, top_p, top_k, max_tokens, seed, stop, frequency_penalty, presence_penalty, and response_format. Values are recorded as supplied to the model. A parameter that was left at the provider default MAY be omitted; an implementation that wishes to record the effective default MAY include it.

* *prompt* (object, OPTIONAL): The prompt material presented to the model. At most one of the following content members SHOULD be present; hash MAY accompany either:

  - *text* (string): The full rendered prompt as a single string.

  - *messages* (array): The prompt as an ordered array of role or content objects, for chat-style models. The structure is recorded as presented to the model.

  - *template* (string): An identifier or URL for a prompt template, used when the prompt is generated from a versioned template rather than recorded inline.

  - *hash* (string): A content hash, in the form defined in Section 4.3, over the canonical prompt. Present when the prompt text is withheld (for privacy or size) but must remain verifiable, or alongside inline text as an integrity check.

When the prompt contains personal data and the vCon may be redistributed, implementations SHOULD record hash and omit text and messages. See Section 7.

* *inputs* (array, OPTIONAL): The vCon elements that were supplied to the model as part of the prompt - the derivation of the generated content. Each entry is an object:

  - *element* (string, REQUIRED): One of "dialog", "analysis", or "attachment".

  - *index* (integer, REQUIRED): The index of the referenced entry in the corresponding top-level array.

  - *content_hash* (string, OPTIONAL): A content hash, in the form defined in Section 4.3, over the referenced element’s content as it was given to the model. Binding inputs by hash makes the derivation verifiable even if the referenced element is later redacted or amended.

* *output_hash* (string, OPTIONAL): A content hash, in the form defined in Section 4.3, over the generated content (the analysis or dialog body this record describes). Binds the provenance record to the specific output it explains.

* *software* (string, OPTIONAL): The harness, application, or pipeline that performed the generation, distinct from the model vendor (for example "vconic-summarizer/2.1"). Analogous to the recording_agent of [I-D.draft-howe-vcon-agent-session].

* *registry* (object, OPTIONAL): A pointer to an external transparency service holding an attestation of this provenance record, for audit trails.

  - *type* (string, REQUIRED): The registry protocol. This document defines the value "scitt".

  - *url* (string, REQUIRED): The endpoint of the transparency service. When type is "scitt", the URL references a SCITT Transparency Service implementing SCRAPI [I-D.draft-ietf-scitt-scrapi].

4.3. Content Hashing

Every hash token defined by this extension - prompt.hash, inputs[].content_hash, and output_hash - uses the same form as the content_hash parameter of [I-D.draft-ietf-vcon-vcon-core]: the hash algorithm name, a hyphen, and the base64url encoding (without padding) of the digest, for example sha512-q2dGq.... Implementations SHOULD use SHA-512 (sha512-).

Hashes are computed over the exact byte sequence presented to or produced by the model. When the hashed material is a JSON value (for example a structured messages prompt or a JSON analysis body), implementations SHOULD canonicalize it with the JSON Canonicalization Scheme [RFC8785] before hashing, so that independent parties compute the same digest.

4.4. Example: Analysis with Provenance

A summary analysis, recording the model, parameters, a withheld prompt bound by hash, the dialog it was derived from, and a hash of the summary itself:

```
{
  "type": "summary",
  "dialog": [0],
  "vendor": "anthropic",
  "product": "claude-opus-4-8",
  "schema": "https://example.com/schemas/call-summary/v3",
  "encoding": "json",
  "body": "{\"summary\":\"Customer requested a refund...\"}",
  "provenance": {
    "model": {
      "vendor": "anthropic",
      "name": "claude-opus-4-8",
      "version": "claude-opus-4-8-20260115"
    },
    "generated_at": "2026-06-04T14:22:05Z",
    "parameters": {
      "temperature": 0.2,
      "top_p": 0.95,
      "max_tokens": 1024,
      "seed": 42
    },
    "prompt": {
      "template": "https://example.com/prompts/call-summary/v3",
      "hash": "sha512-3qFxLp8Yc0r2..."
    },
    "inputs": [
      {
        "element": "dialog",
        "index": 0,
        "content_hash": "sha512-7dInK2pQ9..."
      }
    ],
    "output_hash": "sha512-9aZ0bN4mE...",
    "software": "vconic-summarizer/2.1",
    "registry": {
      "type": "scitt",
      "url": "https://transparency.example.com/entries"
    }
  }
}
```

4.5. Example: Generated Dialog with Provenance

A synthesized agent reply placed in the dialog array, with provenance recorded on the dialog object:

```
{
  "type": "text",
  "start": "2026-06-04T14:20:00Z",
  "parties": [1],
  "mediatype": "text/plain",
  "body": "I have started your refund; it will post in 3-5 days.",
  "encoding": "none",
  "provenance": {
    "model": {
      "vendor": "openai",
      "name": "gpt-x"
    },
    "generated_at": "2026-06-04T14:20:00Z",
    "parameters": {
      "temperature": 0.7
    },
    "prompt": {
      "hash": "sha512-bQ1wRt5..."
    },
    "inputs": [
      { "element": "dialog", "index": 0 },
      { "element": "dialog", "index": 1 }
    ],
    "output_hash": "sha512-Kd9..."
  }
}
```

5. Processing Requirements

5.1. Producing Provenance

An implementation that generates content and records its provenance:

* MUST set model.vendor, model.name, and generated_at.

* SHOULD set parameters to the decoding parameters it actually used.

* SHOULD record prompt, as inline material when permitted, or as a hash when the prompt must be withheld.

* SHOULD populate inputs with references to every vCon element it supplied to the model, and SHOULD bind each by content_hash.

* SHOULD set output_hash over the generated content.

* When the record is on an analysis object, SHOULD set model.vendor and model.name consistently with the analysis vendor and product parameters.

5.2. Consuming Provenance

An implementation that processes a provenance record:

* MAY verify output_hash against the content of the object that carries the record, and SHOULD treat a mismatch as an integrity failure for that generated content.

* MAY verify each inputs[].content_hash against the referenced element when that element is still present in the vCon. A consumer MUST NOT treat a missing or redacted input element as a verification failure on its own; the absence is expected after redaction (see Section 6).

* MUST validate that each inputs[].index is a valid index into the corresponding top-level array before dereferencing it.

* MUST ignore members of parameters it does not recognize.

5.3. Reference Validation

inputs references are validated the same way as other vCon index references: an element/index pair MUST identify an existing entry in the named top-level array at the time the reference is written. After redaction or amendment, a previously valid reference MAY dangle; consumers SHOULD use the accompanying content_hash, where present, to confirm the original input rather than relying on the index.

6. Lifecycle, Redaction, and Transparency

Provenance records interact with the lifecycle extension [I-D.draft-howe-vcon-lifecycle] in two ways.

First, a prompt.text or prompt.messages value can contain personal data drawn from the conversation. When a redacted form of the vCon is produced for broader distribution, these prompt fields SHOULD be removed and replaced by prompt.hash, preserving verifiability without re-exposing the underlying data.

Second, because inputs[].content_hash and output_hash bind a generation to specific inputs and a specific output, they let a verifier confirm a derivation even across a redaction boundary: the hashes attest to what was used and what was produced, while the underlying content may be removed.

These properties make a provenance record a natural subject for a transparency service. A vCon (or an individual analysis entry) carrying a provenance record can be signed and its statement submitted to a SCITT Transparency Service [I-D.draft-ietf-scitt-scrapi]; the resulting receipt attests not merely that an analysis exists, but that a named model, under recorded parameters, produced a specific output from specific inputs. The optional registry member records where such an attestation can be found.

7. Security Considerations

Recording provenance expands the information carried in a vCon, and some of that information is sensitive.

* *Prompts can carry personal data.* A rendered prompt frequently embeds conversation content, retrieved records, and identifiers. Inline prompt.text or prompt.messages therefore inherit the privacy sensitivity of the source conversation and MUST be governed by the same lawful basis [I-D.draft-howe-vcon-lawful-basis] and redaction controls. When in doubt, record prompt.hash and omit the inline prompt.

* *Prompts can carry secrets.* System prompts and templates may contain proprietary instructions, API keys, or other credentials. Implementations MUST NOT record secrets in prompt or parameters; where a system prompt is proprietary, record prompt.template or prompt.hash rather than the text.

* *Provenance is an assertion, not proof of origin.* A provenance record is written by the producing party and, by itself, is not cryptographic evidence that a particular model produced the content. Trust in the record derives from the JWS signature over the vCon as defined in [I-D.draft-ietf-vcon-vcon-core], and, where present, from a transparency receipt over the signed statement. A consumer that needs assurance of origin MUST rely on those mechanisms, not on the presence of the record alone.

* *Hash binding is only as good as the canonicalization.* Verifiers and producers MUST agree on the canonical form of hashed material (see Section 4.3); a divergence in canonicalization yields spurious mismatches. JSON material SHOULD be canonicalized per [RFC8785].

* *Reproducibility is not implied.* Recording seed and decoding parameters does not guarantee that re-running the model reproduces the output, since provider-side model versions and infrastructure change. The record documents how the content was generated, not a recipe guaranteed to regenerate it.

8. IANA Considerations

8.1. vCon Extensions Names Registry

This document registers the following value in the vCon Extensions Names Registry established by [I-D.draft-ietf-vcon-vcon-core]:

* *Extension Name*: provenance

* *Extension Description*: Generation provenance for content produced by a generative model, carried on the analysis or dialog object it describes.

* *Change Controller*: IESG

* *Specification Document(s)*: RFC XXXX

8.2. vCon Analysis Object Parameter Names Registry

This document registers the following value in the vCon Analysis Object Parameter Names Registry established by [I-D.draft-ietf-vcon-vcon-core]:

* *Parameter Name*: provenance

* *Parameter Description*: Generation provenance for the analysis content (model, decoding parameters, prompt, inputs, output hash).

* *Change Controller*: IESG

* *Specification Document(s)*: RFC XXXX

8.3. vCon Dialog Object Parameter Names Registry

This document registers the following value in the vCon Dialog Object Parameter Names Registry established by [I-D.draft-ietf-vcon-vcon-core]:

* *Parameter Name*: provenance

* *Parameter Description*: Generation provenance for a machine-generated dialog turn (model, decoding parameters, prompt, inputs, output hash).

* *Change Controller*: IESG

* *Specification Document(s)*: RFC XXXX

8.4. vCon Provenance Registry Type Values Registry

This document requests IANA to establish a new registry for the values of the registry.type member of a provenance record, with the following initial registration:

* *Type Value*: scitt

* *Description*: A transparency service implementing the SCITT (Supply Chain Integrity, Transparency, and Trust) protocol.

* *Change Controller*: IESG

* *Specification Document(s)*: RFC XXXX, [I-D.draft-ietf-scitt-scrapi]

Registration Template:

Type Value: The string value used as the registry type identifier.

Description: Brief description of the registry type and its purpose.

Change Controller: For Standards Track RFCs, list "IESG". For others, give the name of the responsible party.

Specification Document(s): Reference to defining documents with URIs where available.

9. References

9.1. Normative References

[I-D.draft-ietf-vcon-vcon-core]
Petrie, D. G., "The JSON format for vCon - Conversation Data Container", Work in Progress, Internet-Draft, draft-ietf-vcon-vcon-core-02, January 2026, <https://datatracker.ietf.org/doc/draft-ietf-vcon-vcon-core/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3339] Klyne, G., "Date and Time on the Internet: Timestamps", July 2002, <https://www.rfc-editor.org/rfc/rfc3339.html>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

9.2. Informative References

[C2PA] Coalition for Content Provenance and Authenticity, "Coalition for Content Provenance and Authenticity (C2PA) Technical Specification", 2024, <https://c2pa.org/specifications/>.

[EU-AI-ACT] European Union, "Regulation (EU) 2024/1689 (Artificial Intelligence Act)", 2024, <https://artificialintelligenceact.eu/>.

[I-D.draft-howe-vcon-agent-session] McCarthy-Howe, T., "vCon Agent Session", 2026, <https://datatracker.ietf.org/doc/draft-howe-vcon-agent-session/>.

[I-D.draft-howe-vcon-lawful-basis] McCarthy-Howe, T., "vCon Lawful Basis", 2026, <https://datatracker.ietf.org/doc/draft-howe-vcon-lawful-basis/>.

[I-D.draft-howe-vcon-lifecycle] McCarthy-Howe, T., "vCon Lifecycle", 2026, <https://datatracker.ietf.org/doc/draft-howe-vcon-lifecycle/>.

[I-D.draft-ietf-scitt-scrapi] Birkholz, H., "SCITT Reference REST API", 2025, <https://datatracker.ietf.org/doc/draft-ietf-scitt-scrapi/>.

[I-D.draft-ietf-vcon-overview] McCarthy-Howe, T., "The vCon - Conversation Data Container - Overview", 2025, <https://datatracker.ietf.org/doc/draft-ietf-vcon-overview/>.

[NIST-AI-RMF]
National Institute of Standards and Technology, "AI Risk Management Framework 1.0", January 2023, <https://www.nist.gov/itl/ai-risk-management-framework>.

[RFC8785] Rundgren, A., "JSON Canonicalization Scheme (JCS)", June 2020, <https://www.rfc-editor.org/rfc/rfc8785.html>.

Acknowledgments

This extension exists only because of the broader vCon community, whose discussions on recording how generated content comes to be shaped the design throughout. Thanks to the vCon Working Group as a whole for its feedback and guidance on extension design patterns, and in particular:

* Brian Rosen and Chris Wendt, the vCon Working Group chairs, for shepherding the working group and its drafts.

* Daniel Petrie, for making the concept of a conversation container real and for the core specification this extension builds on.

* Henk Birkholz, whose work on SCITT transparency and on Verifiable Agent Conversations informed the derivation-and-attestation model at the heart of this document.

* Allistair Woodman, for connecting the first dots between vCon and SCITT.

* Jeff Pulver and Cody Launius, for their continued collaboration and support of the vCon effort.

* Andy Newton, for careful review of related vCon work.

* Vinnie Micciche, for unwavering support of vCons in general.

* Pavan Kumar, for review and contributions across the extension family.

This extension reuses vocabulary from the companion agent session and lawful basis extensions. Thanks to everyone in the vCon community whose questions made the case for giving generated content a verifiable record of how it came to be.

Author’s Address

Thomas McCarthy-Howe
VCONIC
United States
Email: ghostofbasho@gmail.com